Балтийский государственный технический университет «ВОЕНМЕХ»

им. Д. Ф. Устинова

Кафедра И5

«Информационные системы и программная инженерия»

Практическое задание № 2

по дисциплине «Программирование на ЯВУ»

на тему «События в SDL 2.0»

Выполнил:

Студент Дубровский В.И.

Группа И582

Преподаватель:

Спирин Д.О.

Санкт-Петербург

2019

*Цель работы:*

Изучить основные события библиотеки SDL2.0, разобраться с проектированием простейших приложений с графическим интерфейсом с использованием классов.

*Задание:*

1. В файле l2.c приведен текст программы: разобраться и устранить возможные ошибки, проанализировать виды событий и причины их срабатывания.
2. В программе перевести все сообщения на русский язык.
3. Изменить логику построения и структуру программы взяв за основу ЛР 1.
4. На основе полученных знаний написать программу, создающую два окна. В первом фиксируются движение мыши по экрану, во втором выводятся координаты положения курсора и наоборот. В каждом окне имеется кнопка, включающая/отключающая слежение за мышью с выводом соответствующего сообщения.
5. Реализовать возможность перемещения кнопки в пределах окна.

*Выполнение:*

В программе были обнаружены и устранены следующие ошибки:

1. 164. event.window.windowID = SDL\_GetWindowID(param);, param — void\* явно не приводится к SDL\_Window\*
2. 177. SDL\_HideWindow(param);, param — void\* явно не приводится к SDL\_Window\*

Каждое из событий было проанализировано, каждое событие было связано с определенным событием SDL2. Все сообщения были переведены на русский.

Взяв за основу 1 лабораторную работу, программа была написана в объектно-ориентированном стиле.

*Текст программы:*

main.cpp

#include <iostream>

#include "Application.h"

constexpr auto TITLE = "Laboratornay rabota 2. Remake.";

constexpr auto WIDTH = 500;

constexpr auto HEIGHT = 500;

Application application(TITLE, WIDTH, HEIGHT);

int main(int argc, char\* argv[])

{

try {

return application.run();

}

catch (std::exception& e)

{

std::cerr << e.what() << std::endl;

application.destroy();

return 1;

}

}

EventFilter.h

#pragma once

#include <SDL.h>

#include <stdlib.h>

void fillScreen(SDL\_Window\* window);

int asmFunction(void);

int eventFilter(void\* userdata, SDL\_Event\* event);

Uint32 repeatOnceFunction(Uint32 interval, void\* param);

Uint32 customEventFunction(Uint32 interval, void\* param);

EventFilter.cpp

#include "EventFilter.h"

void fillScreen(SDL\_Window\* window) {

SDL\_Surface\* screen = SDL\_GetWindowSurface(window);

SDL\_FillRect(screen, NULL, SDL\_MapRGB(screen->format, rand() % 255, rand() % 255, rand() % 255));

SDL\_UpdateWindowSurface(window);

}

int asmFunction(void) {

static int internalValue = 1;

#ifdef \_\_GNUC\_\_

\_\_asm\_\_("movl %0, %%eax\n\t"

"add %%eax, %0"

: "=r" (internalValue)

: "r" (internalValue));

#elif \_MSC\_VER

\_asm {

mov eax, internalValue

add internalValue, eax

};

#endif

return internalValue;

}

int eventFilter(void\* userdata, SDL\_Event\* event) {

switch (event->type) {

case SDL\_KEYDOWN:

if (event->key.keysym.sym == SDLK\_q && event->key.keysym.mod == KMOD\_LCTRL) {

SDL\_Event exitEvent = { SDL\_QUIT };

SDL\_PushEvent(&exitEvent);

}

SDL\_Log("Кнопочка вниз/key Down %d", event->key.keysym.sym);

break;

case SDL\_KEYUP:

SDL\_Log("Кнопочка вверх/key Up %d", event->key.keysym.sym);

break;

case SDL\_TEXTEDITING:

SDL\_Log("Клавиатурка редактирует текст/Keyboard text editing (composition). Composition is '%s', cursor start from %d and selection lenght is %d", event->edit.text, event->edit.start, event->edit.length);

break;

case SDL\_TEXTINPUT:

SDL\_Log("Ввод текста с клавиатуры/Keyboard text input. Text is '%s'", event->text.text);

break;

case SDL\_FINGERMOTION:

SDL\_Log("Пальчик/Finger: %lld, x: %f, y: %f", event->tfinger.fingerId, event->tfinger.x, event->tfinger.y);

break;

case SDL\_FINGERDOWN:

SDL\_Log("Пальчик вниз/Finger: %lld down - x: %f, y: %f",

event->tfinger.fingerId, event->tfinger.x, event->tfinger.y);

return 1;

case SDL\_FINGERUP:

SDL\_Log("Пальчик вверх/Finger: %lld up - x: %f, y: %f", event->tfinger.fingerId, event->tfinger.x, event->tfinger.y);

break;

case SDL\_MULTIGESTURE:

SDL\_Log("Множественный жест/Multi Gesture: x = %f, y = %f, dAng = %f, dR = %f", event->mgesture.x, event->mgesture.y, event->mgesture.dTheta, event->mgesture.dDist);

SDL\_Log("Множественнный жест: циферкаснизукоснуться/ Multi Gesture: numDownTouch = %i", event->mgesture.numFingers);

break;

case SDL\_DOLLARGESTURE:

SDL\_Log("Жест номер исполнен, ошибка/Gesture %lld performed, error: %f", event->dgesture.gestureId, event->dgesture.error);

break;

case SDL\_DOLLARRECORD:

SDL\_Log("Записанный жест/Recorded gesture: %lld", event->dgesture.gestureId);

break;

case SDL\_MOUSEMOTION:

SDL\_Log("Мышку подвигали/Mouse Move. X=%d, Y=%d, Относительно/RelativeX=%d, Относительно/RelativeY=%d", event->motion.x, event->motion.y, event->motion.xrel, event->motion.yrel);

break;

case SDL\_MOUSEBUTTONDOWN:

if (event->button.button == SDL\_BUTTON\_LEFT)

asmFunction();

SDL\_Log("Мышечка кнопка вниз/Mouse Button Down %u", event->button.button);

break;

case SDL\_MOUSEBUTTONUP:

SDL\_Log("Мышечка кнопка вверх/Mouse Button Up %u", event->button.button);

break;

case SDL\_MOUSEWHEEL:

SDL\_Log("Мышечка колёсико/Mouse Wheel X=%d, Y=%d", event->wheel.x, event->wheel.y);

break;

case SDL\_QUIT:

SDL\_Log("Пользовательский-запрос выйти/User-requested quit");

return 1;

case SDL\_WINDOWEVENT:

switch (event->window.event) {

case SDL\_WINDOWEVENT\_SHOWN:

SDL\_Log("Окошко номер показано/Window %d shown", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_HIDDEN:

SDL\_Log("Окошко номер спрятано/Window %d hidden", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_EXPOSED:

fillScreen(SDL\_GetWindowFromID(event->window.windowID));

SDL\_Log("Окошко номер обнаружено/Window %d exposed", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_MOVED:

SDL\_Log("Окошко номер передвинуто ->/Window %d moved to %d,%d", event->window.windowID, event->window.data1, event->window.data2);

break;

case SDL\_WINDOWEVENT\_RESIZED:

SDL\_Log("Окошко номер изменено ->/Window %d resized to %dx%d", event->window.windowID, event->window.data1, event->window.data2);

break;

case SDL\_WINDOWEVENT\_SIZE\_CHANGED:

SDL\_Log("Окошко номер размер изменён ->/Window %d size changed to %dx%d", event->window.windowID, event->window.data1, event->window.data2);

break;

case SDL\_WINDOWEVENT\_MINIMIZED:

SDL\_Log("Окошко номер минимизировано/Window %d minimized", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_MAXIMIZED:

SDL\_Log("Окошко номер максимизировано/Window %d maximized", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_RESTORED:

SDL\_Log("Окошко номер Window %d restored", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_ENTER:

SDL\_Log("Мышка вошла в окно/Mouse entered window %d", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_LEAVE:

SDL\_Log("Мышка покинуло окно/Mouse left window %d", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_FOCUS\_GAINED:

SDL\_Log("Окно номер было под фокусом клавиатуры/Window %d gained keyboard focus", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_FOCUS\_LOST:

SDL\_Log("Окно номер потеряла фокус клавиатуры/Window %d lost keyboard focus", event->window.windowID);

break;

case SDL\_WINDOWEVENT\_CLOSE:

SDL\_Log("Окно номер закрыто/Window %d closed", event->window.windowID);

break;

default:

SDL\_Log("Окно номер получило неизвестный эвент/Window %d got unknown event %d", event->window.windowID, event->window.event);

break;

}

break;

default:

SDL\_Log("Получили неизвестный эвент/Got unknown event %d", event->type);

break;

}

return 0;

}

Uint32 customEventFunction(Uint32 interval, void\* param) {

SDL\_Event event = { SDL\_WINDOWEVENT };

SDL\_Log("Timer signaled with interval %d ms", interval);

event.window.windowID = SDL\_GetWindowID((SDL\_Window\*)param);

event.window.event = SDL\_WINDOWEVENT\_EXPOSED;

SDL\_PushEvent(&event);

return(interval);

}

Uint32 repeatOnceFunction(Uint32 interval, void\* param) {

SDL\_Event exitEvent = { SDL\_QUIT };

SDL\_Log("Timer signaled with interval %d ms", interval);

if (asmFunction() != 0) {

SDL\_HideWindow((SDL\_Window\*)param);

SDL\_ShowSimpleMessageBox(SDL\_MESSAGEBOX\_INFORMATION, "Something going wrong", "Find me! I'm scared", NULL);

SDL\_Delay(15000); /\* 15 sec \*/

SDL\_LogWarn(SDL\_LOG\_CATEGORY\_APPLICATION, "You didn't find me! You disappointed me... I'm leaving.");

SDL\_PushEvent(&exitEvent);

}

return 0;

}

Application.h

#pragma once

#include<SDL.h>

#include<memory>

#include<vector>

class Application

{

private:

bool \_isRunning = false;

SDL\_Window\* \_window = nullptr;

SDL\_Renderer\* \_renderer = nullptr;

SDL\_Event \_events;

SDL\_TimerID repeatOnceFunctionTimer;

SDL\_TimerID customEventFunctionTimer;

void init();

void render();

void setTimer(SDL\_TimerID timer, SDL\_TimerCallback callback, void\* param);

public:

Application() = delete;

explicit Application(const char\* title, int w, int h);

~Application();

void destroy();

int run();

};

Application.cpp

#include "Application.h"

#include <iostream>

#include <SDL\_ttf.h>

#include "EventFilter.h"

#include <exception>

Application::Application(const char\* title, int w, int h)

{

if(SDL\_Init(SDL\_INIT\_EVERYTHING) == -1) std::exception(static\_cast<const char\*>(SDL\_GetError()));

if (TTF\_Init() == -1) throw std::exception(static\_cast<const char\*>(TTF\_GetError()));

\_window = SDL\_CreateWindow(title, SDL\_WINDOWPOS\_CENTERED, SDL\_WINDOWPOS\_CENTERED, w, h, SDL\_WINDOW\_SHOWN);

if (\_window) std::cerr << "Ok Window" << std::endl;

else std::exception(static\_cast<const char\*>(SDL\_GetError()));

\_renderer = SDL\_CreateRenderer(\_window, -1, NULL);

if (\_renderer) std::cerr << "Ok Renderer" << std::endl;

else std::exception(static\_cast<const char\*>(SDL\_GetError()));

}

Application::~Application()

{

}

void Application::destroy()

{

SDL\_RemoveTimer(repeatOnceFunctionTimer);

SDL\_RemoveTimer(customEventFunctionTimer);

SDL\_DestroyRenderer(\_renderer);

SDL\_DestroyWindow(\_window);

SDL\_Quit();

TTF\_Quit();

}

void Application::init()

{

\_isRunning = true;

}

void Application::render()

{

SDL\_SetRenderDrawColor(\_renderer, 0, 0, 0, SDL\_ALPHA\_OPAQUE);

SDL\_RenderClear(\_renderer);

SDL\_RenderPresent(\_renderer);

}

int Application::run()

{

SDL\_SetEventFilter(eventFilter, nullptr);

if (!(customEventFunctionTimer = SDL\_AddTimer(2000 /\* 2 sec \*/, customEventFunction, \_window))) {

SDL\_ShowSimpleMessageBox(SDL\_MESSAGEBOX\_ERROR, "Error", "Unable to create custom event timer. See the log for more info.", \_window);

SDL\_LogCritical(SDL\_LOG\_CATEGORY\_APPLICATION, "Unable to create custom event timer, error: %s", SDL\_GetError());

}

if (!(repeatOnceFunctionTimer = SDL\_AddTimer(10000 /\* 10 sec \*/, repeatOnceFunction, \_window))) {

SDL\_ShowSimpleMessageBox(SDL\_MESSAGEBOX\_ERROR, "Error", "Unable to create repeat once timer. See the log for more info.", \_window);

SDL\_LogCritical(SDL\_LOG\_CATEGORY\_APPLICATION, "Unable to create repeat once timer, error: %s", SDL\_GetError());

}

SDL\_WaitEvent(NULL);

destroy();

return 0;

}

Программа работает и выполняет те же функции, как и приведенная в вложеном файле l2.c

На основе полученных знаний была написана программа для пункта 4.

Основные классы приложения:

1. BaseApp — базовый класс для всех графических элементов приложения. Задаёт единый интерфейс и поля.
2. Window — класс реализующий абстракцию окна. От него наследуются окна, специализацией которых занимается разработчик.
3. Signal — класс содержащий в себе массив указателей на функции и метод их вызова.
4. CallBack — класс реализующий статические методы связывания обратного вызова методов
5. ErrorLoger — класс, который наследуется от std::exception. Основной класс исключений приложения.
6. Fonts — класс отвечающий за работу с файлами, в конкретном случае с шрифтами.
7. Widget — базовый класс для виджетов приложения.
8. Label — класс реализующий надписи.
9. Button — класс реализующий кнопку.
10. Application – класс реализующий основную логику приложения. От него наследуются производные классы приложений, специализацией которых занимается разработчик.

По заданию было сделана возможность передвижения кнопки по нажатию правой кнопки мыши. Результат работы программы представлен на рисунках 1-2.



Рисунок 1 — Демонстрация работы программы

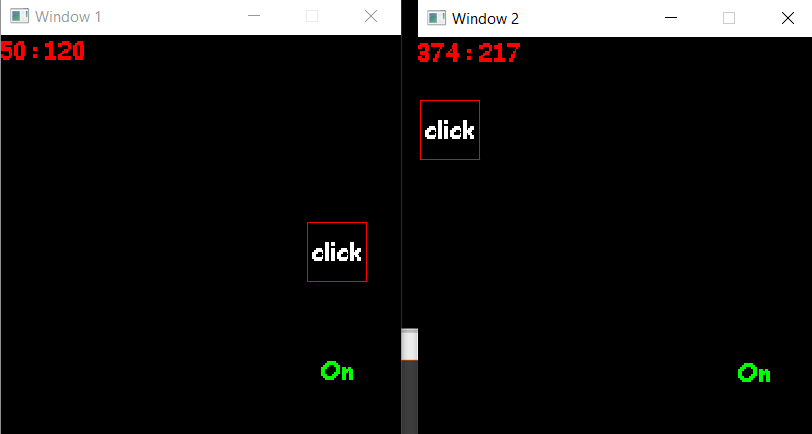


Рисунок 2 — Демонстрация передвинутых кнопок